IN THE CLAIMS

1. (Original) A method, comprising:

identifying a region of a main thread that likely has one or more delinquent loads, the one or more delinquent loads representing loads which likely suffer cache misses during an execution of the main thread;

analyzing the region for one or more helper threads with respect to the main thread; and

generating code for the one or more helper threads, the one or more helper threads being speculatively executed in parallel with the main thread to perform one or more tasks for the region of the main thread.

2. (Original) The method of claim 1, wherein identifying the region comprises:

generating one or more profiles for cache misses of the region; and

analyzing the one or more profiles to identify one or more candidates for thread-based prefetch operations.

3. (Original) The method of claim 2, wherein generating one or more profiles comprises:

executing an application associated with the main thread with debug information; and

sampling cache misses and accumulating hardware counter for each static load of the region to generate the one or more profiles for each cache hierarchy.

4. (Currently Amended) The method of claim ~~2~~3, wherein analyzing the one or more profiles comprises:

correlating the one or more profiles with respective source code based on the debug information; and identifying top loads that contribute cache misses above a predetermined level as the delinquent loads.

5. (Original) The method of claim 1, wherein analyzing the region comprises:

building a dependent graph that captures data and control dependencies of the main thread; and

performing a slicing operation on the main thread based on the dependent graph to generate the helper threads.

6. (Original) The method of claim 5, wherein analyzing the region further comprises:

performing a scheduling between the main thread and the helper threads; and

determining a communication scheme between the main thread and the helper threads.

7. (Original) The method of claim 6, wherein analyzing the region further comprises determining a synchronization period for the helper threads to synchronize the main thread and the helper threads, each of the helper threads performing its tasks within the synchronization period.

8. (Currently Amended) A machine-readable <u>storage</u> medium having executable code to cause a machine to perform a method, the method comprising:

identifying a region of a main thread that likely has one or more delinquent loads, the one or more delinquent loads representing loads which likely suffer cache misses during an execution of the main thread;

analyzing the region for one or more helper threads with respect to the main thread; and generating code for the one or more helper threads, the one or more helper threads being speculatively executed in parallel with the main thread to perform one or more tasks for the region of the main thread.

9. (Currently Amended) The machine-readable storage medium of claim 8, wherein identifying the region comprises:

generating one or more profiles for cache misses of the region; and

analyzing the one or more profiles to identify one or more candidates for thread-based prefetch operations.

10. (Currently Amended) The machine-readable storage medium of claim 9, wherein generating one or more profiles comprises:

executing an application associated with the main thread with debug information; and

sampling cache misses and accumulating hardware counter for each static load of the region to generate the one or more profiles for each cache hierarchy.

11. (Currently Amended) The machine-readable storage medium of claim ~~9~~10, wherein analyzing the one or more profiles comprises:

correlating the one or more profiles with respective source code based on the debug information; and

identifying top loads that contribute cache misses above a predetermined level as the delinquent loads.

12. (Currently Amended) The machine-readable storage medium of claim 8, wherein analyzing the region comprises:

building a dependent graph that captures data and control dependencies of the main thread; and performing a slicing operation on the main thread based on the dependent graph to generate the helper threads.

13. (Currently Amended) The machine-readable <u>storage</u> medium of claim 12, wherein analyzing the region further comprises:

performing a scheduling between the main thread and the helper threads; and

determining a communication scheme between the main thread and the helper threads.

14. (Currently Amended) The machine-readable <u>storage</u> medium of claim 13, wherein analyzing the region further comprises determining a synchronization period for the helper threads to synchronize the main thread and the helper threads, each of the helper threads performing its respective tasks within the synchronization period.

15. (Original) A data processing system, comprising:

a processor capable of performing multi-threading operations;

a memory coupled to the processor; and

a process executed by the processor from the memory to cause the processor to

identify a region of a main thread that likely has one or more delinquent loads, the one or more delinquent loads representing loads which likely suffer cache misses during an execution of the main thread,

analyze the region for one or more helper threads with respect to the main thread, and

generate code for the one or more helper threads, the one or more helper threads being speculatively executed in parallel with the main thread to perform one or more tasks for the region of the main thread.

16. (Original) The data processing system of claim 15, wherein the process is executed by a compiler during a compilation of an application.

17. (Currently Amended) A method, comprising:

executing a main thread of an application in a multi-threading system; and

spawning one or more helper threads from the main thread to perform one or more computations for the main thread when the main thread enters a region having one or more delinquent loads, code of the one or more helper thread being created <u>separately from code of the main thread</u> during a compilation of the main thread.

18. (Original) The method of claim 17, further comprising:

creating a thread pool to maintain a list of thread contexts; and

allocating one or more thread contexts from the thread pool to generate the one or more helper threads.

19. (Original) The method of claim 18, further comprising:

terminating the one or more helper threads when the main thread exits the region; and

releasing the thread contexts associated with the one or more helper threads back to the thread pool.

20. (Original) The method of claim 17, further comprising determining a time period for each of the helper threads, each of the helper threads being terminated when the respective time period expires.

21. (Original) The method of claim 20, wherein each of the helper threads terminates when the time period expires even if the respective helper thread has not been accessed by the main thread.

22. (Original) The method of claim 17, further comprising discarding results generated by the one or more helper threads when the main thread exits the region, the results not being reused by another region of the main thread.

23. (Currently Amended) A machine-readable storage medium having executable code to cause a machine to perform a method, the method comprising:

executing a main thread of an application in a multi-threading system; and

spawning one or more helper threads from the main thread to perform one or more computations for the main thread when the main thread enters a region having one or more delinquent loads, code of the one or more helper thread being created separately from code of the main thread during a compilation of the main thread.

24. (Currently Amended) The machine-readable storage medium of claim 23, wherein the method further comprises:

creating a thread pool to maintain a list of thread contexts; and

allocating one or more thread contexts from the thread pool to generate the one or more helper threads.

25. (Currently Amended) The machine-readable storage medium of claim 24, wherein the method further comprises:

terminating the one or more helper threads when the main thread exits the region; and releasing the thread contexts associated with the one or more helper threads back to the thread pool.

26. (Currently Amended) The machine-readable storage medium of claim 23, wherein the method further comprises determining a time period for each of the helper threads, each of the helper threads being terminated when the respective time period expires.

27. (Currently Amended) The machine-readable storage medium of claim 26, wherein each of the helper threads terminates when the time period expires even if the respective helper thread has not been accessed by the main thread.

28. (Currently Amended) The machine-readable storage medium of claim 23, wherein the method further comprises discarding results generated by the one or more helper threads when the main thread exits the region, the results not being reused by another region of the main thread.

29. (Currently Amended) A data processing system, comprising:

a processor capable of performing multi-threading operations;

a memory coupled to the processor; and

a process executed by the processor from the memory to cause the processor to execute a main thread of an application in a multi-threading system, and spawn one or more helper threads from the main thread to perform one or more computations for the main thread when the main thread enters a region having one or more delinquent loads, code of the one or more helper thread being created separately from code of the main thread during a compilation of the main thread.

30. (Canceled)